

Processor Identification and Feature Determination

33

When writing software intended to run on several different types of Intel Architecture processors, it is generally necessary to identify the type of processor present in a system and the processor features that are available to an application. This chapter describes how to identify the processor that is executing the code and determine the features the processor supports. It also shows how to determine if an FPU or NPX is present. See Chapter 17, *Intel Architecture Compatibility*, in the *Intel Architecture Software Developer's Manual, Volume 3*, for a complete list of the features that are available for the different Intel Architecture processors.

33.1 Processor Identification

The CUID instruction returns the processor type for the processor that executes the instruction. It also indicates the features that are present in the processor, including the existence of an on-chip FPU. The following information can be obtained with this instruction:

- The highest operand value the instruction responds to (2 for the Pentium® Pro processors and 1 for the Pentium processors and recent Intel486™ processors).
- The processor's family identification (ID) number, model ID, and stepping ID.
- The presence of an on-chip FPU.
- Support for or the presence of the following architectural extensions and enhancements:
 - Virtual-8086 mode enhancements.
 - Debugging extensions.
 - Page-size extensions.
 - Read time stamp counter (RDTSC) instruction.
 - Read model specific registers (RDMSR) and write model specific registers (WRMSR) instructions.
 - Physical address extension.
 - Machine check exceptions.
 - Compare and exchange 8 bytes instruction (CMPXCHG8B).
 - On-chip, advanced programmable interrupt controller (APIC).
 - Memory-type range registers (MTRRs).
 - Page global flag.
 - Machine check architecture.
 - Conditional move instruction (CMOVcc).
 - MMX™ technology.
- Cache and TLB information.

To use this instruction, a source operand value of 0, 1 or 2 is placed in the EAX register. Processor identification and feature information is then returned in the EAX, EBX, ECX, and EDX registers. See “CPUID—CPU Identification” in Chapter 3 of the *Intel Architecture Software Developer's Manual, Volume 2*, for more detailed information about the instruction.

AP-485, *Intel Processor Identification and the CPUID Instruction* (Order Number 241618), provides additional information and example source code for use in identifying Intel Architecture processors. It also contains guidelines for using the CPUID instruction to help maintain the widest range of software compatibility. The following guidelines are among the most important, and should always be followed when using the CPUID instruction to determine available features:

- Always begin by testing for the “GenuineIntel,” message in the EBX, EDX, and ECX registers when the CPUID instruction is executed with EAX equal to 0. If the processor is not genuine Intel, the feature identification flags may have different meanings than are described in “CPUID—CPU Identification” in Chapter 3 of the *Intel Architecture Software Developer's Manual, Volume 2*.
- Do not assume a value of 1 in a feature identification flag indicates that a given feature is present. For future feature identification flags, a value of 1 may indicate that the specific feature is not present.
- Test feature identification flags individually and do not make assumptions about undefined bits.

Note that the CPUID instruction will cause the invalid opcode exception (#UD) if executed on a processor that does not support it. The CPUID instruction application note provides a code sequence to test the validity of the CPUID instruction. Also, this test code (for CPUID valid) is not reliable when executed in virtual-8086 mode. To avoid this, if the test code is written to run in real-address mode, the SMSW instruction must be used to read the PE bit from the MSW (lower half of CR0). If PE flag is set to 1, the Real Mode code is actually being executed in virtual-8086 mode, and the test sequence cannot be guaranteed to return reliable information. (Note that the new version of the CPUID application note (AP-485, *Intel Processor Identification and the CPUID Instruction* (Order Number 241618-005)), explains this virtual-8086 problem, but the older versions of the application note do not.)

33.2 Identification of Earlier Intel Architecture Processors

The CPUID instruction is only available in the Pentium Pro, Pentium, and recent Intel486 processors. For the earlier Intel Architecture processors (including the earlier Intel486 processors), several other architectural features can be exploited to identify the processor.

The settings of bits 12 and 13 (IOPL), 14 (NT), and 15 (reserved) in the EFLAGS register (see Figure 27-7) is different for Intel's 32-bit processors than for the Intel 8086 and Intel 286 processors. By examining the settings of these bits (with the PUSHF/PUSHFD and POP/POPFD instructions), an application program can determine whether the processor is an 8086, Intel286, or one of the Intel 32-bit processors:

- 8086 processor — Bits 12 through 15 of the EFLAGS register are always set.
- Intel 286 processor — Bits 12 through 15 are always clear in real-address mode.
- 32-bit processors — In real-address mode, bit 15 is always clear and bits 12 through 14 have the last value loaded into them. In protected mode, bit 15 is always clear, bit 14 has the last

value loaded into it, and the IOPL bits depends on the current privilege level (CPL). The IOPL field can be changed only if the CPL is 0.

Other EFLAG register bits that can be used to differentiate between the 32-bit processors:

- Bit 18 (AC) — Implemented only on the Pentium® Pro, Pentium, and Intel486™ processors. The inability to set or clear this bit distinguishes an Intel386 processor from the other Intel 32-bit processors.
- Bit 21 (ID) — Determines if the processor is able to execute the CUID instruction. The ability to set and clear this bit indicates that the processor is a Pentium Pro, Pentium, or later version Intel486 processor.

To determine whether an FPU or NPX is present in a system, applications can write to the FPU/NPX status and control registers using the FNINIT instruction and then verify the correct values are read back using the FNSTENV instruction.

After determining that an FPU or NPX is present, its type can then be determined. In most cases, the processor type will determine the type of FPU or NPX; however, an Intel386 processor is compatible with either an Intel 287 or Intel 387 math coprocessor. The method the coprocessor uses to represent ∞ (after the execution of the FINIT, FNINIT, or RESET instruction) indicates which coprocessor is present. The Intel 287 math coprocessor uses the same bit representation for $+\infty$ and $-\infty$; whereas, the Intel 387 math coprocessor uses different representations for $+\infty$ and $-\infty$.

